Proposal

Domain Background

I want to be a real estate investor, so I thought working on a real estate related project would be beneficial. Since I haven't been in the ML field for too long, I picked a project from kaggle: House Prices: Advanced Regression Techniques. Link for this exercise: https://www.kaggle.com/c/house-prices-advanced-regression-techniques Armed with my ML knowledge and 80 features in the provided dataset, I am looking to accomplish reasonable errors in my housing price predictions. Through this exercise, I would be able to have a sense of what impact home prices beyond my intuition.

I would also note that the Ames Housing dataset was complied by Dean De Cock for use in data science education. This is a modernized and expanded version of the often cited Boston Housing dataset. Please use the link to access the Journal of Statistics Education article published in 2011, detailing the Ames Housing dataset and Regression Project: https://ww2.amstat.org/publications/jse/v19n3/decock.pdf

Problem Statement

The goal of this exercise is to predict house prices using available features. Thisis a supervised regression exercise. Kaggle suggested Random Forest and Gradient Boosting as ways to tackle this exercise. However, I think the exercise is much more comprehensive than plugging train and test datasets into sklearn. There will be data exploration, PCA, grid search, and MLP components. More quantifiable justifications for this exercise selection:

- Quantifiable: With different ML techniques, we can express how to find housing prices in math or logical terms.
- Measurable: We will mainly measure the effectiveness of the model by root mean square error of the datasets (for the provided test dataset, we'll split it into 3: train, validation and test).
- Replicable: With models pickled or parameters saved, we can certainly replicate the model and the result.

Datasets and Inputs

The dataset for this exercise originated from the Ames City Assessor's Office. Variables that required special domain knowledge and variables that were present for weighing and adjustment purposes were removed. The data contains 79 features (23 nominal, 23 ordinal, 14 discrete and 20 continuous), 1 target, and 1459 observations. Since sensitive information, such as address or zip codes, were never present, it will be difficult to improve model performance by introducing external data and joining with the existing dataset. Please refer to the attached features dictionary excel file.

The goal of the exercise is to predict SalePrice, which is the target in our given dataset. It is a continuous variable and describe the final sales price of the property. I will explore all 79 features (23 nominal, 23 ordinal, 14 discrete and 20 continuous) and their potentials to predict SalePrice. From my initial exploration, features such as LotArea is highly correlated with the SalesPrice. Here are some thoughts on how I will treat different categories of variables:

Variable Type	**Nominal**	**Ordinal**	**Discrete**	**Continous**
Example Feature	Heating (type of heating)	OverallQual (rating for the material and finish of the property)	FullBath (number of full bathrooms in the property)	LotArea (property plot size)
Potential Treatment	Onehot encoding	See if transformation is necessary	See if transformation is necessary	Apply log transformation and normalization

Because data from this exercise came from kaggle, I'll be using the given training dataset as my train, validation, and test datasets. I will split the data into train v test (80:20, random split), then split the remaining data further into train and validation (80:20) for my model optimization process. Note that train and validation will be used for individual model optimization, while test will be use to select a best performing model.

Solution Statement

The solution will be house price predictions, which is continuous. Because of the selected ML, I will be able to quantify the solution in math or logical terms. I will be able to evaluate our predictions with the validation and test datasets to calculate the error measurement (root mean square error) and replicate the solution because the optimal model will be preserved. Keep in mind, this is an exercise to minimize the error term on the prediction/solution. Here's a list of algorithms and techniques I want to try:

1. Data exploration, including feature engineering and PCA:
    i. Visualize distributions for each feature, and see if transformation (numpy.log(x + 1)) or normalization (sklearn.preprocessing.MinMaxScaler) are required.
    ii. Find outliers and determine if they need to be removed.
    iii. one-hot encode the ordinal or nominal features (pandas.get_dummies())
    iv. Will use sklearn.decomposition.PCA to discover dimensions about the data best maximize the features' variance.
2. Supervised model selections leveraging grid search/k-folds including:
    i. DecisionTreeRegressor (benchmark)
    ii. RandomForestRegressor

iii. AdaBoostRegressor

iv. KNeighborsRegressor

v. GradientBoostingRegressor

vi. MLPRegressor

vii. KerasRegressor

Benchmark Model

During my model optimization exercise, I will use the results from DecisionTreeRegressor as my benchmark. All the model performance will be compared to the DecisionTreeRegressor, given they are fitted to the same train and validation set.

Evaluation Metrics

The main evaluation metric will be root mean square error, which is represented by:

$$\text{RMSD} = \sqrt{\frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{n}}$$

Root mean square error is a standard evaluation for modeling exercises with target that is continuous. I will split the provided dataset to do train, validation and test sets. I will choose the modeling hyperparameters based on how the root mean squared error performed on train and validation datasets. After optimizing the hyperparameters, I will then choose the models based on test data performance.

I can also time the modeling process so we can also use duration as an evaluation metric.

Project Design

To complete the exercise, here's the steps that I will take:

1. Explore data – I will take a look at how the data is distributed by looking at the max, min, median and, and histogram. I can also take a look at how the features correlate by using the numpy.corrcoef() or pandas.corr() functions.
2. Clean data – I will look at how each feature is presented, and how it relates to the target. I will also look at how the outliers and missings data should be managed. Visualization at this stage is needed as visualization for outliers, correlations and r square make more compelling stories.
    i. Outliers: Use a transformation. They should only be removed unless it is recorded erroneously, or creating a significant association. Otherwise, we can try run the model with or without the outlier to see how the model fits to the test dataset.

ii. Missing data: the simplest way to deal with missing data is to remove the observation. However, if this significantly reduce the observations, I will use sklearn.preprossing.Imputer to try to fill those missing values.

3. Prepare data – I will use sklearn.model_selection.train_test_split to randomly split the data into train and test sets.

4. Feature treatments – with 79 features, perhaps PCA and feature engineering/transformation are required. I'll take a look at how these methodologies could help reduce the dimensionality of this exercise. Note that different model may require different feature treatments. Different algorithms may consider features differently. Also note that I will fit models to an untreated dataset. Future feature engineering/models performances should also be compared to these benchmarks to see if feature engineering is value-added.

5. Model selection – given this is a regression exercise, I will experiment with different modeling algorithms. To optimize each model's hyperparameters, I will use grid search in choosing hyper parameters. During each optimization exercise, I will look at how the error terms improve as well as if the models are overfitting. I will use visualization to illustrate how I reach various decisions along the exercise. I will compare each optimized model and select a final model. The model's output on the provided test dataset will be submitted to kaggle for evaluation. Here are some models that I'll consider:

   i. DecisionTreeRegressor – use GridSearchCV, experiment with different depths and cross-validation to optimize the model performance.

   ii. RandomForestRegressor – use GridSearchCV, experiment with different depths and cross-validation to optimize the model performance

   iii. AdaBoostRegressor – use GridSearchCV, experiment with different loss and cross-validation to optimize the model performance.

   iv. KNeighborsRegressor – use GridSearchCV, experiment with different neighbors, algos and cross-validation to optimize the model performance.

   v. GradientBoostingRegressor – use GridSearchCV, experiment with different max_depth, lossand n_estimators and cross-validation to optimize the model performance.

   vi. MLPRegressor – use GridSearchCV, experiment with different hidden_layer_sizes, activation, learning_rate, momentum.

   vii. KerasRegressor – use kfold, experiment with compile optimizers, activation, nb_epochs and batch_size, number of layers, etc.

   viii. Other possibilities: XGBoost, Naive Bayes, and SVM.